



INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : H04L 9/28	A1	(11) International Publication Number: WO 94/10778 (43) International Publication Date: 11 May 1994 (11.05.94)
(21) International Application Number: PCT/US93/10585	(81) Designated States: AT, AU, BB, BG, BR, BY, CA, CH, CZ, DE, DK, ES, FI, GB, HU, JP, KP, KR, KZ, LK, LU, MG, MN, MW, NL, NO, NZ, PL, PT, RO, RU, SD, SE, SK, UA, VN, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).	
(22) International Filing Date: 2 November 1993 (02.11.93)		
(30) Priority data: 970,611 3 November 1992 (03.11.92) US		
(71) Applicant: NOVELL, INC. [US/US]; 122 East 1700 South, Provo, UT 84601 (US).	Published <i>With international search report. Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.</i>	
(72) Inventor: KINGDON, Kevin ; 1331 East 600 North, Orem, UT 84057 (US).		
(74) Agents: HECKER, Gary, A. et al.; Hecker & Harriman, 2049 Century Park East, Suite 1200, Los Angeles, CA 90067 (US).		
(54) Title: METHOD AND APPARATUS FOR AUTHENTICATION OF CLIENT SERVER COMMUNICATION		
(57) Abstract	<p>The present invention provides a method and apparatus for message packet authentication to prevent the forging of message packets. A portion of the message digest, referred to as the signature, is then appended to the actual message when it is sent over the wire (205). The receiving station strips the signature from the message (208), prepends the same secret session key (209) and creates its own message digest (212). The signature of the digest created by the receiving station is compared to the signature of the digest appended by the sending station (213). If there is a match, an authentic message is assumed (214). If there is no match, the message is considered as invalid and discarded (216).</p>	
	<pre> graph TD A[Create message] --> B[Prepend session key] B --> C{64 bytes?} C -- No --> D[Send message] C -- Yes --> E[Put with ds] E --> F[Compute MD] F --> G[Append signature to message] G --> H[Send message] H -- client to server --> I[Receive message] I --> J[Strip signature] J --> K[Prepend session key to message] K --> L[Compute MD Create Digest] L --> M{64 bytes?} M -- No --> N[Put with ds] N --> O[Compute MD Create Digest] O --> P{Signature = Signature?} P -- Yes --> Q[Valid message] P -- No --> R[Unauthenticated message discard] </pre>	

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	GB	United Kingdom	MR	Mauritania
AU	Australia	GE	Georgia	MW	Masai
BB	Barbados	GN	Guinea	NE	Niger
BE	Belgium	GR	Greece	NL	Netherlands
BP	Burkina Faso	HU	Hungary	NO	Norway
BG	Bulgaria	IE	Ireland	NZ	New Zealand
BJ	Benin	IT	Italy	PJ	Poland
BR	Brazil	JP	Japan	PT	Portugal
BY	Belarus	KE	Kenya	RO	Romania
CA	Canada	KG	Kyrgyzstan	RU	Russian Federation
CF	Central African Republic	KP	Democratic People's Republic of Korea	SU	Soviet Union
CG	Central African Republic	KR	Republic of Korea	SE	Sweden
CH	Switzerland	KZ	Kazakhstan	SI	Slovenia
CI	Côte d'Ivoire	LI	Liechtenstein	SK	Slovakia
CM	Cameroon	LK	Sri Lanka	SN	Senegal
CN	China	LU	Luxembourg	TD	Chad
CS	Czechoslovakia	LV	Latvia	TO	Togo
CZ	Czech Republic	MC	Monaco	TJ	Tajikistan
DE	Germany	MD	Republic of Moldova	TT	Trinidad and Tobago
DK	Denmark	MG	Madagascar	UA	Ukraine
ES	Spain	ML	Mali	US	United States of America
FI	Finland	MN	Mongolia	UZ	Uzbekistan
FR	France			VN	Viet Nam
GA	Gabon				

METHOD AND APPARATUS FOR AUTHENTICATION OF CLIENT
SERVER COMMUNICATION

5 FIELD OF THE INVENTION

This invention relates to the field of network communications.

BACKGROUND ART

10

Personal computers, or workstations, may be linked through a computer network to allow the sharing of data, applications, files, processing power, communications and other resources, such as printers, modems, mass storage and the like. Generally, the sharing of resources is 15 accomplished by the use of a network server. The server is a processing unit dedicated to managing the centralized resources, managing data and sharing these resources with other PC's and workstations, often referred to as "clients". The server, network and PC's or workstations, combined together, constitute a client/server computer network. An example of a 20 client/server network model is illustrated in Figure 1.

Figure 1 illustrates a client machine 101 coupled to a server machine 102. The client machine 101 may be a PC, workstation, etc. The server machine may be a dedicated processor, PC, workstation, etc, that includes 25 mass storage on which files are stored. Typically, the mass storage is a disk drive or other suitable device.

The client machine 101 is comprised of a client 102 that communicates with a client stub 103. The client stub 103 communicates 30 with a transport entity 104. The server machine 105 includes a server 106, server stub 107, and transport entity 108.

Referring to the client machine 101, the client 102 is a local processor that utilizes files of the server. The client stub 103 is a collection of local 35 procedures that enable the client to access the server. The transport entity 104 provides access to the network, or "wire" 109. Wire 109 refers to the communications medium between the client and server and may be an actual hardwired communications medium, or may be a wireless

connection. Similarly, the server stub 107 is a collection of procedures that enable the server to communicate with the client, and transport entity 108 provides access from the server to the wire 109.

5 In operation, communication between the client and server is in the form of requests (from the client) and replies (from the server). This communication is in the form of remote procedure calls. The client is analogous to an application calling a procedure and getting a result. The difference is that the procedure is not necessarily on the same machine as
10 the client 101, but rather on the server machine 105.

Initially, the client 102 calls a stub procedure located on the client machine in the client stub 103 (resident in the client 102 local address space).
15 The client stub 103 constructs a message from the call and provides it to the transport entity 104. The transport entity 104 communicates the message on the wire 109 to the server machine 105. At the server, the transport entity 108 passes the message to the server stub 107. The server stub then calls the appropriate server procedure from the server 106. The server 106 operates on the message and then returns the procedure and any result to the server stub 107. The server stub 107 constructs a reply message and provides it to the transport entity 108. The reply message is sent to the transport entity 104 of the client machine 101 over the wire 109. The transport entity provides the reply message to the client stub 103. The client stub 103 returns the procedure and any value returned by the server to the client 102.
25

On a computer network, clients and users have different levels of privileges. Certain functions, adding users, deleting users, changing passwords, etc., are restricted to the highest privileged users. These users and clients are often network administrators, and it is necessary for these users to
30 be able to modify the network as necessary. In addition, there may be certain types of files or activities that are restricted from most users. For example, financial data is often restricted to users who have a need to know or use the financial data. Generally, other users are not permitted to access that data.

35 In a client/server model, messages are transported as "packets". An example of a message packet is illustrated in Figure 3A. The message consists of a 4-byte length header (low high) indicator 301. The length

header 301 identifies the length of the message that follows and includes the following information:

5 CheckSum
5 PacketLength
5 TransportControl
5 HPacketType
5 DestinationNet
5 DestinationNode
10 DestinationSocket
10 SourceNet
10 SourceNode
10 SourceSocket

15 The length header 301 is followed by a request code 302. The request code 302 is the particular type of procedure being requested by the client. The request code 302 is followed by data 303. The data 303 may be of variable length.

20 One particular type of message packet is referred to as an "NCP packet", where NCP refers to NetWare Core Protocol. (NetWare is a trademark of Novell, Corporation of Provo, Utah). NetWare is an operating system for network systems. An NCP packet includes the following additional information in the length header:

25 packet type
25 sequence number
25 connection low
25 task
30 connection high

35 The standard portion of the message packet provides source address, destination address and length, among other pieces of information. The NCP portion includes a connection number and a sequence number. The station connection number provides the server with an index into a table of active stations. The server uses the active station table to track information about that station's session, including the station's network address and sequence number.

The connection number is used in part as a security check. When a server receives a request packet, it uses the packet's connection number as an index into its connection table. The request packet's network address must match the network address stored in the connection table entry corresponding to the connection number contained in the request packet. This is one method of validating a request packet.

The sequence number is also used to validate packets. The sequence number is a byte that is maintained by both the server and the client. When the client sends a request packet, that client increments the sequence number. Likewise, when a server receives a request packet, it increments that client's sequence number (stored in the server's connection table). The sequence number wraps around on every 256th request made by the client (because it is one byte in length).

Before incrementing the client's sequence number, the server checks the sequence number against a list of already-received request packets. This check is to ensure that the server does not service duplicate request packets. If the sequence number does not indicate a duplicate request packet, the server checks the request packet's sequence number against the sequence number stored in the server's connection table. If these two numbers are not equal, the server discards the packet.

In spite of these precautions, it is sometimes possible to forge a message packet by detecting the network address, connection station, the station's connection number, and the station's sequence number. Typically, the purpose in forging a message packet is to "imitate" a higher privileged user or client so that the privilege level of the forger can be upgraded. The forger may obtain a more privileged station's connection number by capturing network packets from the communications medium. These are network packets that are sent from a higher privileged station to the server. A forger may capture these packets using a protocol analysis tool.

By obtaining a connection number, a forger may attempt to forge a message by sending a message to the server destination address, using the same station connection number as in the intercepted message. However, that alone is not sufficient to enable an intruder to forge a message. As noted above, the server checks the sequence number and compares it against a list of already-received requests. The sequence number of the new

request should have associated with it the next consecutive sequence number. If not, it is an invalid request and the server discards the packet.

- An intruder may attempt to forge a message by "guessing" at the sequence number. Because the sequence numbers "wrap around" after 256, the intruder need only try to make 256 attempts before the sequence number is found. It should be noted that the intruder does not receive responses from the server, but rather must detect responses from the server or detect if a request issued to the server has been executed (e.g., a change in privilege status for the intruder).

- One possible solution to a network intruder is to monitor network use to detect intruder-type activity. For example, the network could be monitored so that trial and error attempts to provide a correct sequence number are detected. For example, a window could be defined with a certain number of allowed failed tries at providing sequence numbers. A problem is that depending on the size of the window for allowed retries, an intruder could randomly provide a correct sequence number within the window. If the window is made smaller, legitimate transactions might be interrupted when the correct sequence number is not provided by a legitimate user. It is desired to provide a method and apparatus for preventing intruder network access instead of just detecting intruder access.

SUMMARY OF THE PRESENT INVENTION

The present invention provides a method and apparatus for message packet authentication to prevent the forging of message packets. After a message packet is created, a secret session key is prepended to the message, and a message digesting algorithm is executed on the altered message to create a message digest. A portion of the message digest, referred to as the signature, is then appended to the actual message when it is sent over the wire. The receiving station strips the signature from the message, prepends the same secret session key and creates its own message digest. The signature of the digest created by the receiving station is compared to the signature of the digest appended by the sending station. If there is a match, an authentic message is assumed. If there is no match, the message is considered as invalid and discarded. An advantage of the present invention is that the session key is never transmitted over the wire. The receiving station (server) already has the key and uses the key along with the message data to recalculate the message digest upon receiving the packet. The shared secret key (session key) is generated during initiation of the NCP session. In addition, cumulative state information is maintained by both the sending station and the receiving station. This state information is also used to authenticate messages.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a client/server model.

5 Figure 2 is a flow diagram of a message session using the session key
of the present invention.

Figures 3A-3G are diagrams of message structure during a message
session.

10 Figure 4 is a flow diagram of the client state during a message session.

Figure 5 is a flow diagram of the server state during a message session.

15 Figure 6 is a flow diagram illustrating the generation of a session key.

Figure 7 is a flow diagram of a method of authenticating a session key.

20 Figure 8 is a block diagram of a computer system in which the present
invention may be implemented.

Figure 9 is a block diagram of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

A method and apparatus for message authentication is described. In the following description, numerous specific details, such as message type, message length, etc., are provided in detail in order to provide a more thorough description of the present invention. It will be apparent, however, that the present invention may be practiced without these specific details. In other instances, well-known features are not described in detail so as not to unnecessarily obscure the present invention.

The present invention provides a signature with each message that identifies and authenticates the sender. In addition, the invention tracks state information about the session and uses its cumulative effect to help protect and authenticate senders.

The signature scheme of the invention takes advantage of an operation known as "message digesting". Message digesting is a scheme to provide data integrity and detect errors. There are a number of message digesting algorithms available for use, some of which are provided by Rivest, Shamir and Associates (RSA). RSA message digesting algorithms are known as MD2, MD4 and MD5. The preferred embodiment of the present invention utilizes a derivative of message digesting algorithm MD4. The MD4 algorithm is described in RFC 1320, "The MD4 Message-Digest Algorithm", R. Rivest, MIT Laboratory for Computer Science and RSA Data Security, Inc. April 1992, incorporated by reference herein. In the preferred embodiment of the present invention, the padding scheme described herein is used instead of the padding scheme described in the MD4 algorithm. However, the present invention may be used with any suitable padding scheme. In addition, the MD2 and MD5 digesting algorithms may be used as the digesting algorithm.

Other cryptographically secure, one way hashing algorithms may be used instead of a digesting algorithm without departing from the scope and spirit of the present invention. A hash function is a mathematical function that maps values from a large domain into a smaller range. In the preferred embodiment the hash function is such that the results of applying the function to a set of values in the domain will be evenly distributed (and apparently random) over the range. Using the present scheme, encryption of a message can be avoided, saving time, while still providing the benefit of an effective digital authentication signature.

MESSAGE SIGNATURE

The method for creating and utilizing a signature to authenticate a message is illustrated in the flow diagram of Figure 2. At step 200, a message is created by the client. This message is as shown in Figure 3A. The message consists of a 4-byte length header (low high) indicator 301. The length header 301 identifies the length of the message that follows and includes source and destination information. The length header 301 is followed by a request code 10 302. The request code 302 is the particular type of procedure being requested by the client. The request code 302 is followed by data 303. The data 303 may be of variable length.

At step 201 a session key, whose creation is described in detail below, 15 is pre-appended to the message. The prepending of the session key of step 201 of Figure 2 is illustrated in Figure 3B. An 8-byte key 304 is prepended to the message before the length indicator 301.

At decision block 202, the argument "64 bytes?" is made. If the 20 argument is false, meaning the message does not have 64 bytes, the system proceeds to step 203 and the remainder of the message is filled with zeros. This is the padding 305 of Figure 3B. In the preferred embodiment, the message digest algorithm requires 64 bytes for operation so that if necessary, padding 305 (consisting of zeros) is added to the end of the message. The 25 request code and data total 52 bytes, with 4 bytes of the length indicated in 8 bits of the session key for a total of 64 bytes.

After step 203, or if the argument at decision block 202 is true, the 30 system proceeds to step 204. At step 204, the message digest algorithm is executed to generate a message digest from the prepended message. The execution of the message digest algorithm creates the 16-byte message digest of Figure 3C.

At step 205, the first eight bytes, i.e. the signature 306, of the digest is 35 appended to the message. This is illustrated in Figure 3D. The 8-byte signature 306 is added to the end of the actual NCP data. No padding is required for sending the message over the wire. At step 206, the message is sent to the server. Steps 200-206 are executed by the client, and steps 206-216 are executed by the server.

At step 206, the server receives the message. At step 208, the server strips the signature 306 from the message. The signature 306 is stripped from the message as illustrated in Figure 3E.

- 5 At step 209, the server pre-appends the session key generated and stored by the server (which should be the same session key as generated and stored by a valid user) to the message. This is illustrated in Figure 3F where the server session key 304' is prepended to the message. At decision block 210, the argument "64 bytes?" is made. If the argument is false, the system 10 proceeds to step 211 and fills the remainder of the message with zeros, as illustrated by padding 305 of Figure 3F.

- 15 After step 211, or if the argument at decision block 210 is true, the system proceeds to step 212. At step 212, the MD4 algorithm is executed to create a server message digest, referred to here as digest'. This results in the 16-byte message digest of Figure 3G. The first 8 bytes, i.e. signature' 306' of this message digest are then stripped away and compared to the signature 306 that was sent with the message by the client.

- 20 At decision block 213, the argument "Signature = Signature?" is made. This step is to determine if the signature generated by the client is the same as the signature' generated by the server. If the argument at decision block 213 is true, the system proceeds to step 214 and a valid message is indicated. At step 215, the message is executed. If the argument at decision block 212 is false, the system 25 proceeds to step 216. At step 216, an unauthorized message is indicated and the message is discarded. At step 217, an alert is generated to indicate that an unauthenticated message was attempted.

- 30 The preferred embodiment of the present invention utilizes eight bytes of the digest as the signature. Any number of bytes or bits of the digest may be used as the signature without departing from the scope of the present invention.

- 35 In some cases, message packets use the entire data field, precluding the ability to append the eight byte message digest. In typical applications, block size negotiations assume 64 bytes for protocol header information. In reality, most protocol headers consume less than 56 bytes, leaving 8 bytes always available for the 8 byte digest information. If eight bytes of free space are not available, an artificially smaller block size is negotiated so that an 8 byte space can be guaranteed.

STATE TRACKING

In addition to the message signature scheme described above, the present invention also provides a method of message authentication using 5 state information. The nature of the MD4 algorithm is such that it is cumulative, i.e. the message digest function can be executed in stages.

For example, a block of a file of messages can be provided to the message digest algorithm, digested and the next block of file can be read in 10 and the digesting continued. The output state of the execution of the digesting algorithm is used as an input state on the next digesting step. The net effect of executing the message digest algorithm in stages is as if the message digest algorithm were executed on an entire block of information in a single pass. Thus, a cumulative effect can be had by remembering the 15 state at the end of each algorithm execution. The present invention takes advantage of this cumulative effect and state information to provide an additional level of security and authentication.

Both the client and server track and store state information. In the 20 present invention, this state information consists of the most recently generated message digest. A provisional message digest, using the current message digest and new message, is generated and stored by the client and server. When a new message is received, a new message digest is generated and compared to the provisional digest. A match is only 25 possible if the other party had the previous state information. The incorrect starting state information will propagate an error that identifies intruders and forged messages.

Referring to Figure 4, a flow diagram of state tracking for a client is 30 illustrated. At step 401, the current state of the client is the full 16 byte message digest generated from a previous valid message. At step 402, the client creates a new message. At step 403, the client applies the message digesting algorithm to the new message, using the current state (digest) as a starting point to create a provisional state.

35 At step 404, the message is sent to the server. At step 405, the client receives a reply from the server. At step 406, the reply is checked, using the provisional state of step 403. This is accomplished by stripping the server generated digest from the message, applying the message digest algorithm to

-12-

the message (using the provisional state as a starting point) and comparing the first eight bytes of the resulting digest to the first eight bytes of the server generated digest.

5 At decision block 407, the argument "valid reply?" is made. If the argument is false, the system proceeds to step 408 and does not advance the state, since an invalid reply has been received. The next message generated by the client will use the existing current state as the starting point for the digesting algorithm. If the argument at decision block 407 is true, the system 10 proceeds to step 409 and advances the state, that is, it makes the provisional state the current state. That new current state will now be the starting point when applying the digesting algorithm to the next message.

15 In some cases, the client may send a burst of data to the server, or a client request may generate a burst reply. The order of the messages in the packet burst (except for the first packet) is not necessarily fixed. This can create problems in calculating state information, because of the cumulative nature of the digesting algorithm. For packet burst replies, the server calculates the message digest for the first reply packet and uses the same 20 state to calculate message digests for all remaining packets in the burst. In this manner, regardless of the order of the packets after the first packet, state integrity can be maintained.

25 Burst requests are handled the same way. The first packet in a stream is factored in the state of the digest algorithm. Subsequent packets in the burst request use the same state as the first packet as the initial state.

30 A flow diagram illustrating the operation of the server in maintaining state information is illustrated in Figure 5. At step 501, the server receives a message from the client. At decision block 502, the argument "burst?" is made. If the argument is true, the system uses the state of the first packet and returns to step 503. If the argument is false, the system proceeds to step 503 and checks the sequence number. At decision block 504, the argument "repeat sequence number?" is made. This is to identify repeat requests. The 35 digest output generated for request packets is always factored into the digest state. Thus, the prior digest state must be maintained by the server when repeat requests are encountered. If the argument at decision block 504 is true, the system proceeds to step 505 and recomputes a provisional state based on the new message, using the stored previous state.

At decision block 506, the argument "burst?" is made. If the argument is true, the system proceeds to step uses the state of the first packet and returns to step 507. If the argument is false, the system proceeds to step 507. At step 507, the signatures of the message are verified based on the provisional state. At decision block 508, the argument "valid?" is made. If the argument is false, the system discards the message and provides an alert for a potential forged message at step 509. If the argument at decision block 508 is true, the system replies to message at step 510.

If the argument at decision block 504 is false, the system proceeds to decision block 511. At decision block 511, the argument "next consecutive sequence number?" is made. If the argument is false, the system proceeds to step 512, declares the message invalid and discards it. In other words, the sequence number was inappropriate. If the argument at decision block 511 is true, the system proceeds to step 513 and verifies the signatures based on the provisional state that has been maintained.

At decision block 514, the argument "valid?" is made. If the argument is false, the system proceeds to step 509, the message is discarded, and an alert is generated. If the argument is true, the system proceeds to step 515. At step 515, the provisional state is redefined as the current state. At step 516, a new provisional state is created based on the reply. At decision block 517, the argument "burst?" is made. If the argument is true, the system uses the state of the first packet and proceeds to step 518. If the argument is false, the system proceeds to step 518 and replies to the client based on the provisional state that has been calculated.

SESSION KEY

A method of generating a client session key is illustrated in Figure 6. When a user attempts to communicate on the network, the user must first be identified to the server. To initiate a session, the user attempts to log on to the client machine. At step 601, the client requests a challenge from the server machine. The challenge consists of 8 bytes of random numbers. The client then prompts the user for an account name and a password at step 602. When the user enters an account name and password, the client machine determines an object ID associated with the account at step 603. (The object ID is a numeric surrogate key or index that is associated with each account).

At step 604, the client machine uses the password and the object ID to compute a digest using a digest algorithm to generate a 16 byte result referred to here as Digest1. At step 605, the client machine builds a buffer of 5 Digest1, the challenge and, optionally, a text string. In the preferred embodiment, the text string of the present invention is "Authorized NetWare Client". The buffer is padded with 0's if necessary to provide 64 bytes for execution of the digest algorithm.

10 At step 606, the client machine performs a second digest on the buffer (Digest1, the challenge, 0's padding, and, optionally, the text string) to generate a digest of the buffer called Dbuffer. At step 607, the first eight bytes of Dbuffer are stripped and defined as the session key. Although eight bytes are used as the session key in the preferred embodiment of the present 15 invention, any number of bytes or bits may be used without departing from the scope of the invention.

The server also has stored the password, account name, and object ID of the user. The server also has generated the challenge, and stores that 20 value. Using the same steps, the server machine can generate the session key. Thus, the session key is never transmitted over the wire. It is generated from secure information at the client machine and the server machine. In addition, because the session key depends in part on the challenge (a random number) the session key is different for each 25 client/server session.

Although not shown in Figure 6, a response is generated to the challenge after step 604. The response, which is sent over the wire to the server, is generated by a hashing algorithm that is different from the 30 hashing algorithm that is used in steps 605 and 606. If step 604 is accomplished using the MD4 algorithm, then the challenge response can use, for example, the MD5 algorithm and the session key is generated using the MD4 algorithm. Alternatively, the challenge response can be generated using the MD4 algorithm and the session key can be generated using a 35 different algorithm, such as the MD5 algorithm. Any different digesting or hashing schemes can be used as long as there is no mapping of the output of one algorithm to the output of the other algorithm.

-15-

The MD5 algorithm is described in RFC 1321, "The MD5 Message-Digest Algorithm", R. Rivest, MIT Laboratory for Computer Science and RSA Data Security, Inc. April 1992 and incorporated herein by reference.

5 SESSION KEY AUTHENTICATION

Figure 7 illustrates a flow diagram of a method of authenticating a session key. At step 701, a client generates a session key as described in connection with Figure 6. At step 702, a request is sent to the server by the 10 client, using the session key to generate a digest and a signature as described in connection with Figure 2.

At step 703, the server strips the signature from the message of the client, and uses the server's stored copy of the account name, password, and 15 object ID to first generate its version of Digest1, namely Digest1', and then uses Digest1' to generate the server version of the session key, namely session key'. At step 704, the server generates Digest' as described in connection with Figure 2.

20 At decision block 705, the argument "Signature = Signature?" is made. If the argument is false, the system proceeds to step 706 and the server sends a negative ack to the client and the server does not change its state. The server does not initialize its state for a new session. If the argument at decision block 705 is true, the system proceeds to step 707 and 25 the server sends an "OK" acknowledgement to the client. At step 708, the server initializes the client state and stores the session key it has generated. At step 709, the server initializes the server state and stores the session key. The initial state of the client and server is defined to be an initial state documented, for example, in the MD4 standard.

30 The client and server of the present invention may be implemented on any conventional or general purpose computer system. An example of one embodiment of a computer system for implementing this invention is illustrated in Figure 8. A keyboard 810 and mouse 811 are coupled to a bi-directional system bus 818. The keyboard and mouse are for introducing user input to the computer system and communicating that user input to CPU 813. The computer system of Figure 8 also includes a video memory 814, main memory 815 and mass storage 812, all coupled to bi-directional 35 system bus 818 along with keyboard 810, mouse 811 and CPU 813. The mass

storage 812 may include both fixed and removable media, such as magnetic, optical or magnetic optical storage systems or any other available mass storage technology. Bus 818 may contain, for example, 32 address lines for addressing video memory 814 or main memory 815. The system bus 818 5 also includes, for example, a 32-bit data bus for transferring data between and among the components, such as CPU 813, main memory 815, video memory 814 and mass storage 812. Alternatively, multiplex data/address lines may be used instead of separate data and address lines.

10 In the preferred embodiment of this invention, the CPU 813 is a 32-bit microprocessor manufactured by Intel, such as the 80386 or 80486. However, any other suitable microprocessor or microcomputer may be utilized. Main memory 815 is comprised of dynamic random access memory (DRAM). Video memory 814 is a dual-ported video random access memory.

15 One port of the video memory 814 is coupled to video amplifier 816. The video amplifier 816 is used to drive the cathode ray tube (CRT) raster monitor 817. Video amplifier 816 is well known in the art and may be implemented by any suitable means. This circuitry converts pixel data stored 20 in video memory 814 to a raster signal suitable for use by monitor 817. Monitor 817 is a type of monitor suitable for displaying graphic images, and in the preferred embodiment of this invention, has a resolution of approximately 1020 x 832. Other resolution monitors may be utilized in this invention.

25 The computer system described above is for purposes of example only. The present invention may be implemented in any type of computer system or programming or processing environment.

A block diagram of the present invention is illustrated in Figure 9. A 30 message generator 901 is a source of messages from a sending station to a receiving station. In this example, the sending station is a client and the receiving station is a server. The message generator 901 provides a message 902. A session key 904 is stored in session key storage 903. The session key 904 is prepended to the message 902 at summer 905, resulting in appended 35 message 906. Appended message 906 is provided to the digester/buffer block 907, where it is digested and the first eight bytes are used as a signature 908. Signature 908 is combined with message 902 at summer 911, resulting in signed message 912. Signed message 912 is coupled to a receiving station through transmitter/receiver 913.

- State storage 909 stores the current and provisional states of the sending station and provides them on line 910 to the digester/buffer block 907 as needed. A challenge 923 and local password 924 are also provided to 5 the digester/buffer block 907 for generation of the session key 904 on initialization of a session.

The received signed message 914 is separated into signature 916 and message 917 elements at subtractor 915. The message 917 is combined with a 10 session key 920 from session key storage 919 at summer 918, resulting in an appended message 921. Appended message 921 is provided to digester/buffer 922, where it is digested. The first eight bytes of the digest define a signature 927. The signature 927 is provided to compare/authenticate block 928, along with the signature 916 of the received message 914. The compare/signature 15 block compares signature 927 with signature 916. When the signatures match, a valid message is indicated. If the signatures do not match, the message is declared invalid and discarded.

State information for the digesting operation of digester/buffer 922 is 20 provided on line 926 from state storage 925. A challenge 929 and password 930 are also provided to digester/buffer 922 for generation of a session key.

The elements of Figure 9 can be implemented as executable 25 instructions in a processing means.

Thus, a method and apparatus for authentication of client/server communication has been described.

CLAIMS

1. A method of authenticating a message transmitted between a sender and a receiver comprising the steps of:
 - 5 generating a message at said sender;
 - combining a session key with said message to create a first appended message;
 - calculating a first digest of said first appended message;
 - combining a first portion of said first digest with said message to create a transmit message;
 - transmitting said transmit message to said receiver;
 - removing said first portion of said first digest from said transmit message to result in said message;
 - combining said session key with said message to generate a second appended message;
 - calculating a second digest of said second appended message;
 - comparing said said portion of said first digest and a second portion of said second digest;
 - authenticating said message when said first portion of said first digest matches said second portion of said second digest.
2. The method of claim 1 wherein said sender is a client in a client/server network.
- 25 3. The method of claim 1 wherein said receiver is a server in a client/server network.
4. The method of claim 1 wherein said step of calculating a first digest of said first appended message is accomplished by executing a digest algorithm on said first appended message.
- 30 5. The method of claim 4 wherein said digest algorithm is an MD4 digest algorithm.
- 35 6. The method of claim 4 wherein a current state of said sender is used as an initial state when executing said digest algorithm to create said first digest.

-19-

7. The method of claim 6 wherein said current state is used as an initial state when executing said digest algorithm to create said second digest.

8. The method of claim 7 wherein said current state is advanced 5 when an authenticated message is received.

9. The method of claim 8 wherein said current state is not advanced when an authenticated message is not received.

10. The method of claim 1 wherein said session key is generated by the steps of:

- providing a random number sequence challenge to said sender;
- requesting a password from a user of said sender;
- generating a first pass digest from said password;
- 15 combining said first pass digest and said challenge in a buffer;
- generating a buffer digest of said buffer;
- defining said session key as a first number of bytes of said buffer digest.

11. Apparatus for authenticating a message transmitted between a 20 sender and a receiver comprising:

- means for generating a message at said sender;
- means for combining a session key with said message to create a first appended message;
- means for calculating a first digest of said first appended message;
- 25 means for combining said a first portion of said first digest with said message to create a transmit message;
- means for transmitting said transmit message to said receiver;
- means for removing said first portion of said first digest from said transmit message to result in said message;
- 30 means for combining said session key with said message to generate a second appended message;
- means for calculating a second digest of said second appended message;
- means for comparing said first portion of said first digest and a second portion of said second digest;
- 35 means for authenticating said message when said first portion of said first digest matches said second portion of said second digest.

12. The apparatus of claim 11 wherein said sender is a client in a client/server network.

13. The apparatus of claim 11 wherein said receiver is a server in a client/server network.

5 14. The apparatus of claim 11 wherein said first digest of said first appended message is calculated by executing a digest algorithm on said first appended message.

10 15. The apparatus of claim 14 wherein said digest algorithm is an MD4 digest algorithm.

16. The apparatus of claim 14 wherein a current state of said sender is used as an initial state when executing said digest algorithm to create said first digest.

15 17. The apparatus of claim 16 wherein said current state is used as an initial state when executing said digest algorithm to create said second digest.

20 18. The apparatus of claim 17 wherein said current state is advanced when an authenticated message is received.

19. The apparatus of claim 18 wherein said current state is not advanced when an authenticated message is not received.

25 20. The apparatus of claim 11 further including means for generating a session key comprising:
means for providing a random number sequence challenge to said sender;
means for requesting a password from a user of said sender;
30 means for generating a first pass digest from said password;
means for combining said first pass digest and said challenge in a buffer;
means for generating a buffer digest of said buffer;
means for defining said session key as a first number of bytes of said buffer digest.

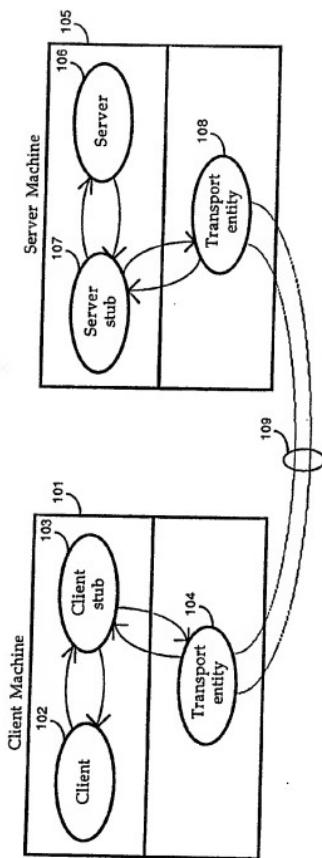
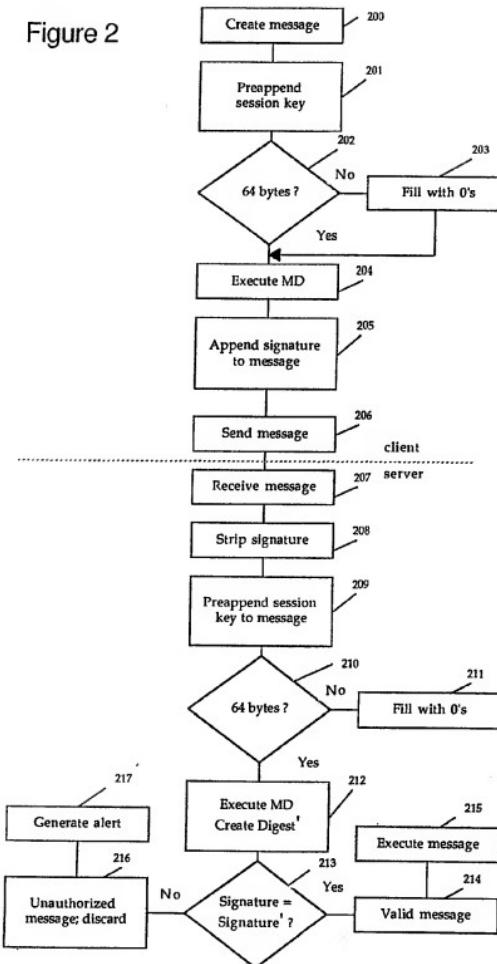


Figure 1

Figure 2



3 / 9

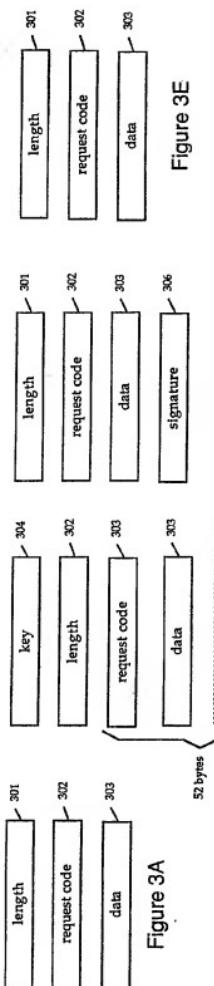
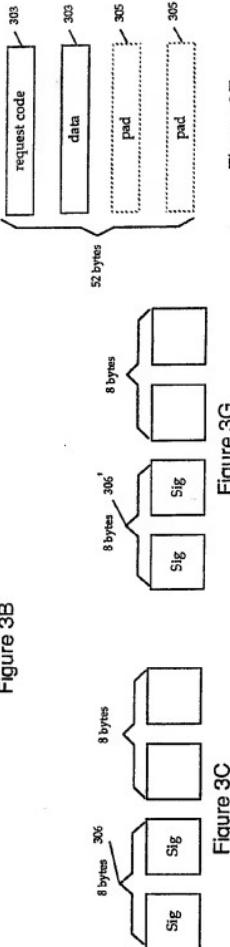
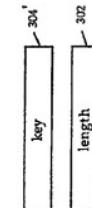


Figure 3E



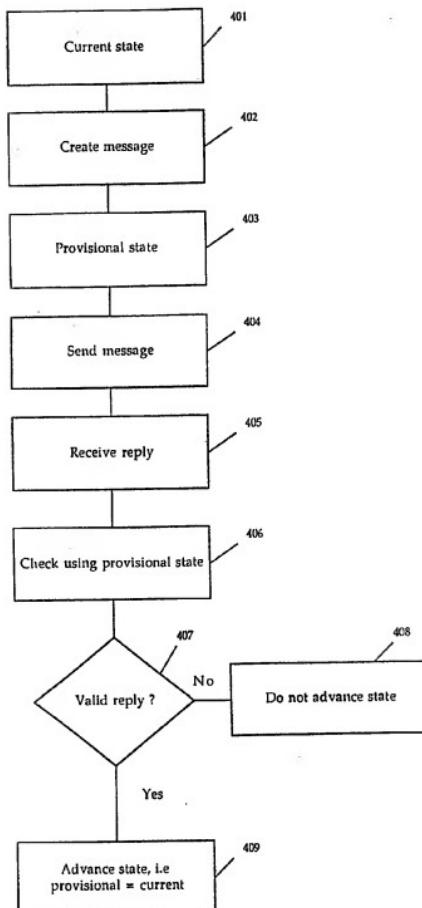


Figure 4

SUBSTITUTE SHEET (RULE 26)

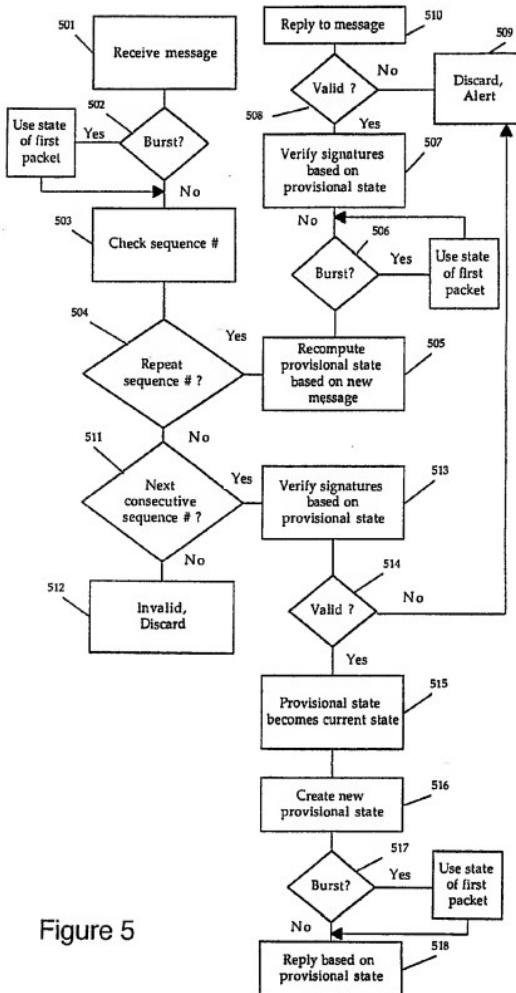


Figure 5

SUBSTITUTED SHEET (RI II F 26)

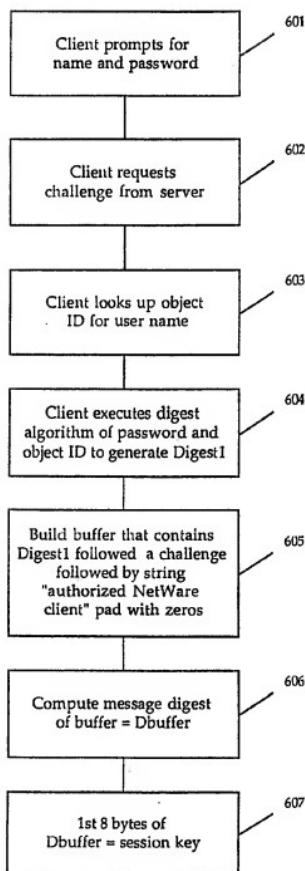


Figure 6

SUBSTITUTE SHEET /R111E 281

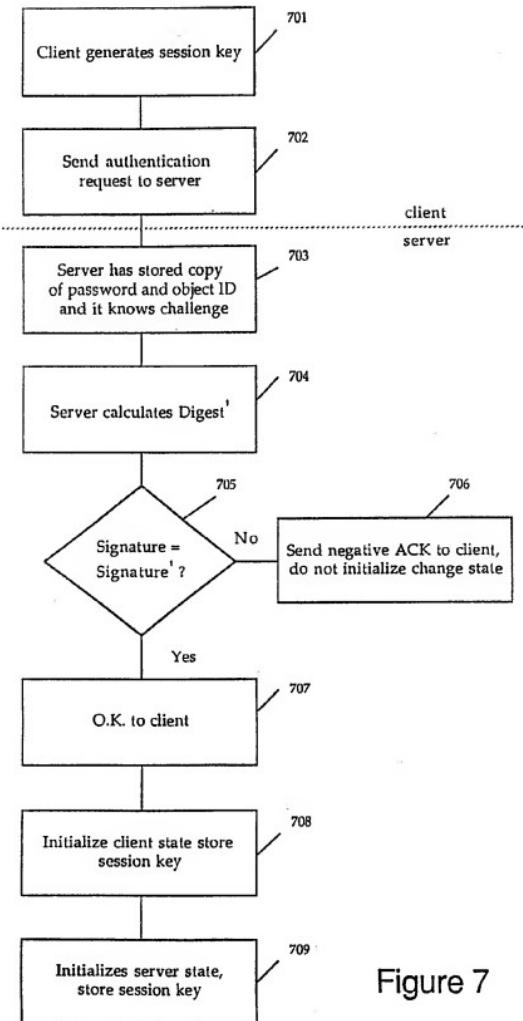


Figure 7

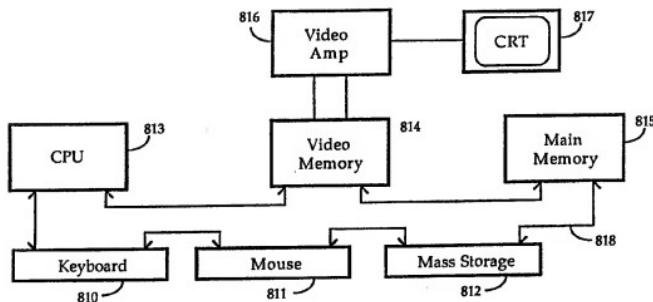
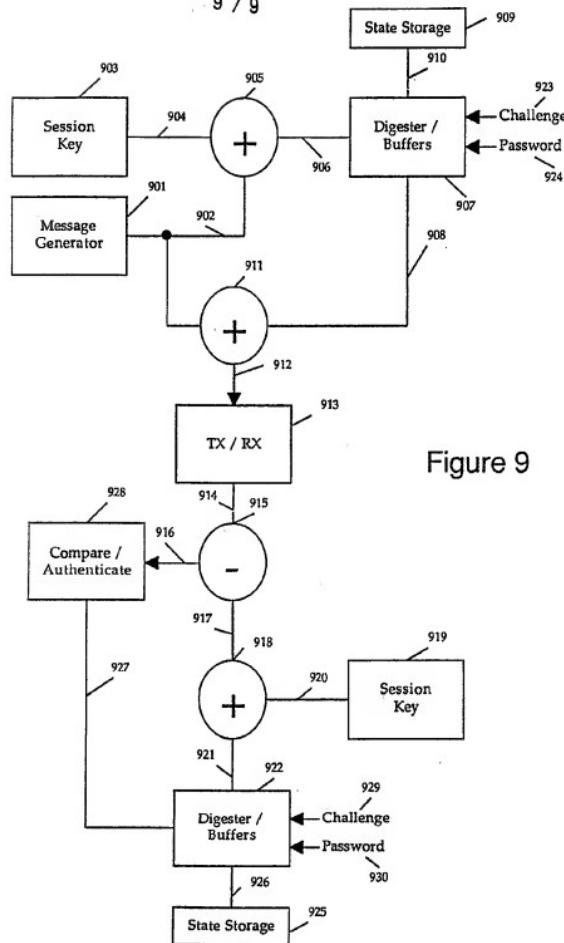


Figure 8

9 / 9



INTERNATIONAL SEARCH REPORT

International application No.
PCT/US93/10585

A. CLASSIFICATION OF SUBJECT MATTER		
IPC(5) :H04L 9/28 US CL :380/28 According to International Patent Classification (IPC) or to both national classification and IPC		
B. FIELDS SEARCHED		
Minimum documentation searched (classification system followed by classification symbols) U.S. : 380/23,25, 28, 30		
Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched		
Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US, A, 4,656,474 (MOLLIER ET AL.) 07 APRIL 1987, SEE ENTIRE DOCUMENT	1-20
Y	US, A, 4,799,258 (DAVIES) 17 JANUARY 1989, SEE COL. 1, LINES 50-65.	1-20
A	US, A, 4,868,877 (FISCHER) 19 SEPTEMBER 1989, SEE FIG. 5	1-20
Y	US, A, 5,050,212 (DYSON) 17 SEPTEMBER 1991, SEE COL. 3, LINES 15-30.	1-20
Y	US, A, 5,140,634 (GUILLOU ET AL.) 18 AUGUST 1992, SEE FIGS. 5 AND 6.	1-20
A,P	US, A, 5,210,795 (LIPNER ET AL.) 11 MAY 1993, SEE FIG. 2.	1-20
<input type="checkbox"/> Further documents are listed in the continuation of Box C. <input type="checkbox"/> See patent family annex.		
<p>* Special categories of cited documents:</p> <ul style="list-style-type: none"> *A* document defining the general state of the art which is not considered to be part of particular relevance *E* earlier document published on or after the international filing date, which may throw doubt on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) *L* document referring to an oral disclosure, use, exhibition or other means *O* document published prior to the international filing date but later than the priority date claimed *T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention *X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone *Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is taken in combination with one or more other such documents, such combination being obvious to a person skilled in the art *Z* document member of the same patent family 		
Date of the actual completion of the international search 13 JANUARY 1994	Date of mailing of the international search report MAR 07 1994	
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer <i>Tracey Fordwyna for</i> SALVATORE CANGIALOSI Telephone No. (703) 308-0482	